Java – Objects, Classes, and Constructors

This session will help you:

- Understand how Java implements OOP.
- Learn to create and manage **objects** and **classes**.
- Work with **constructors** and **the Singleton design pattern**.
- Explore **Java packages**, **import statements**, and **source file declaration rules**.
- Implement a simple case study to apply these concepts.

---

1. Java – Objects & Classes

Objects in Java

An **object** is an instance of a class. It has:

- **State** (defined by instance variables)
- **Behavior** (defined by methods)
- **Identity** (a unique reference in memory)

Example:

```
class Dog {
    String breed;
    int age;
    String color;

    void barking() {
        System.out.println("Woof Woof!");
    }
}

public class TestDog {
    public static void main(String[] args) {
        Dog myDog = new Dog(); // Creating an object
        myDog.breed = "Labrador";
        myDog.age = 3;
        myDog.color = "Brown";
        myDog.barking();
    }
}
```

Classes in Java

A **class** is a blueprint for creating objects. It defines the properties (fields) and behaviors (methods) of objects.

Syntax:

```
class ClassName {
    // Fields (variables)
    // Methods
}
```

Example:

```
class Car {
    String model;
    int year;

    void displayInfo() {
        System.out.println("Model: " + model + ", Year: " + year);
    }
}
```

---

2. Constructors

A **constructor** is a special method used to initialize objects.

- **Has the same name as the class**.
- **Does not have a return type**.
- **Called automatically when an object is created**.

Types of Constructors

1. **Default Constructor** (No parameters)
2. **Parameterized Constructor** (Accepts parameters)

Example:

```
class Student {
    String name;
    int age;

    // Constructor
    Student(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println("Name: " + name + ", Age: " + age);
    }
}

public class TestStudent {
```

```
    public static void main(String[] args) {
        Student s1 = new Student("John", 20);
        s1.display();
    }
}
```

How to Use Singleton Class?

A **Singleton class** ensures that only one instance of the class exists.

Example:

```
class Singleton {
    private static Singleton instance;

    private Singleton() {}

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}

public class SingletonTest {
    public static void main(String[] args) {
        Singleton obj1 = Singleton.getInstance();
        Singleton obj2 = Singleton.getInstance();
        System.out.println(obj1 == obj2); // true
    }
}
```

---

3. Accessing Class Members

Creating an Object

To create an object, use the `new` keyword:

```
ClassName objectName = new ClassName();
```

Example:

```
Car myCar = new Car();
```

Accessing Instance Variables and Methods

Use the **dot operator (`.`)**:

```
myCar.model = "Toyota";
myCar.displayInfo();
```

---

4. Source File Declaration Rules

- The **file name must match the public class name**.
- A Java program can have **only one public class** per file.
- The **main method** must be inside a class.

---

5. Java Package

A **package** is a namespace that organizes related classes and interfaces.

Packages help:

- Avoid naming conflicts between classes.
- Group related classes for better code organization.
- Control access with package-private visibility.

How to Create a Package

1. **Declare the package at the top of your Java file:**

   ```
   package mypackage;
   ```

2. **Save the file in a folder matching the package name:**

   - For `package mypackage;`, save your file inside a folder named `mypackage`.

3. **Compile the class:**

   - From the parent directory, run:

     ```
     javac mypackage/MyClass.java
     ```

4. **Run the class (if it has a main method):**

   - From the parent directory, run:

     ```
     java mypackage.MyClass
     ```

Example: Creating and Using a Package

Step 1: Create a folder named `mypackage`.

Step 2: Inside `mypackage`, create `HelloPackage.java`:

```
package mypackage;

public class HelloPackage {
    public void greet() {
        System.out.println("Hello from mypackage!");
    }
}
```

Step 3: In the parent directory, create `TestPackage.java`:

```
import mypackage.HelloPackage;

public class TestPackage {
    public static void main(String[] args) {
        HelloPackage hp = new HelloPackage();
        hp.greet();
    }
}
```

Step 4: Compile both files from the parent directory:

```
javac mypackage/HelloPackage.java TestPackage.java
```

Step 5: Run the test:

```
java TestPackage
```

Output:

```
Hello from mypackage!
```

Using Packages

Import classes from a package using the `import` statement:

```
import mypackage.HelloPackage;
```

Or import all classes:

```
import mypackage.*;
```

Note: The package declaration must be the first statement (excluding comments) in your Java file.

---

6. Import Statements

Importing a specific class:

```
import java.util.Scanner;
```

Importing all classes from a package:

```
import java.util.*;
```

---

7. A Simple Case Study

Bank Account Management System

This example demonstrates **objects, classes, constructors, and methods**.

```
class BankAccount {
    private String owner;
    private double balance;

    // Constructor
    public BankAccount(String owner, double balance) {
        this.owner = owner;
        this.balance = balance;
    }

    // Deposit method
    public void deposit(double amount) {
        balance += amount;
    }

    // Withdraw method
    public void withdraw(double amount) {
        if (amount <= balance) {
```

```
            balance -= amount;
        } else {
            System.out.println("Insufficient funds");
        }
    }

    // Display balance
    public void displayBalance() {
        System.out.println(owner + "'s balance: " + balance);
    }
}

public class BankDemo {
    public static void main(String[] args) {
        BankAccount account = new BankAccount("Alice", 1000);
        account.deposit(500);
        account.withdraw(300);
        account.displayBalance();
    }
}
```

Expected Output:

```
Alice's balance: 1200.0
```